17

LECTURE 17

Graphs

INTRODUCTION TO GRAPHS

- **What are Graphs**?
  - A class of discrete structures useful for representing relations among objects.
- Graphs are discrete structures consisting of **vertices** and **edges** that connect these vertices.
- There are different kinds of graphs, depending on whether edges have directions, whether multiple edges can connect the same pair of vertices, and whether loops are allowed.

GRAPHS

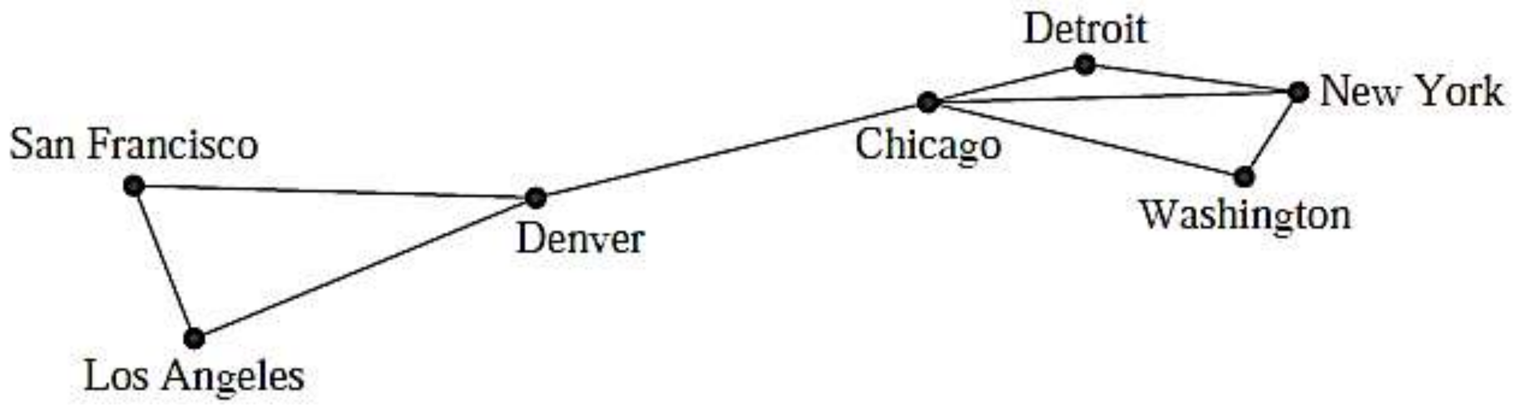
- A graph ***G = (V , E)*** consists of ***V***, a nonempty set of *vertices* (or *nodes*) and ***E***, a set of *edges*.
- Each edge has either one or two vertices associated with it, called its *endpoints*.
- An edge is said to connect its endpoints.

TYPES OF GRAPHS

- Suppose that a network is made up of **data centers** and **communication links** between computers.
- We can represent the location of each data center by a point and each communications link by a line segment.
- This computer network can be modeled using a graph in which the vertices of the graph represent the data centers and the edges represent communication links.
- **The key point is that the way we draw a graph is 'arbitrary'**, as long as the correct connections between vertices are depicted.

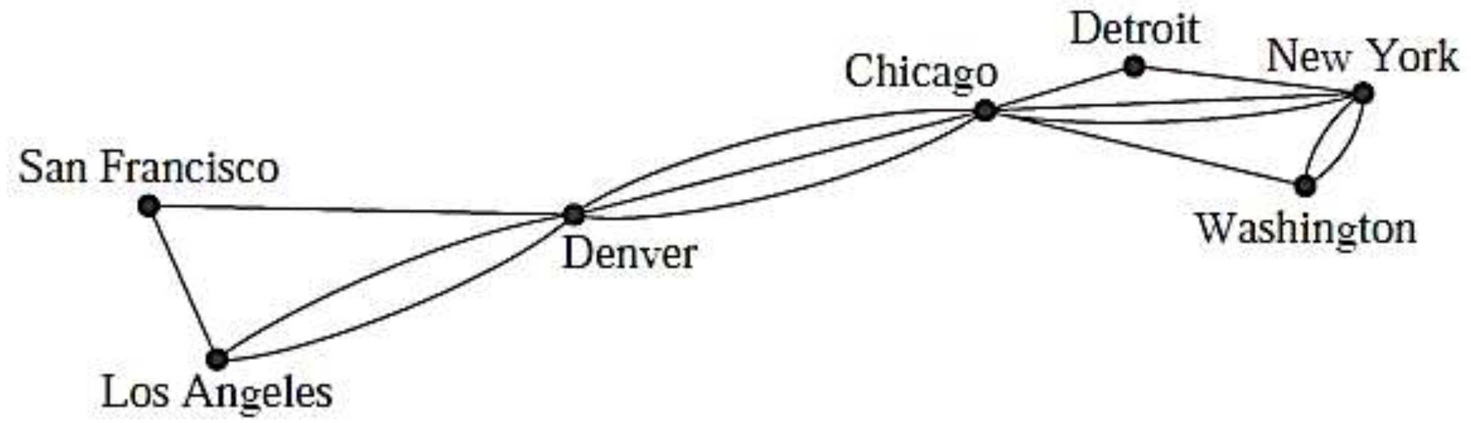
1. SIMPLE GRAPH

- Note that each edge of the graph representing this computer network connects two different vertices. That is, **no edge connects a vertex to itself**.
- Furthermore, **no two different edges connect the same pair of vertices**.
- A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called a **simple graph**.
- Note that in a simple graph, each edge is associated to an unordered pair of vertices, and no other edge is associated to this same edge.
- Simple graphs are **undirected graphs**. Their edges are also said to be **undirected.**



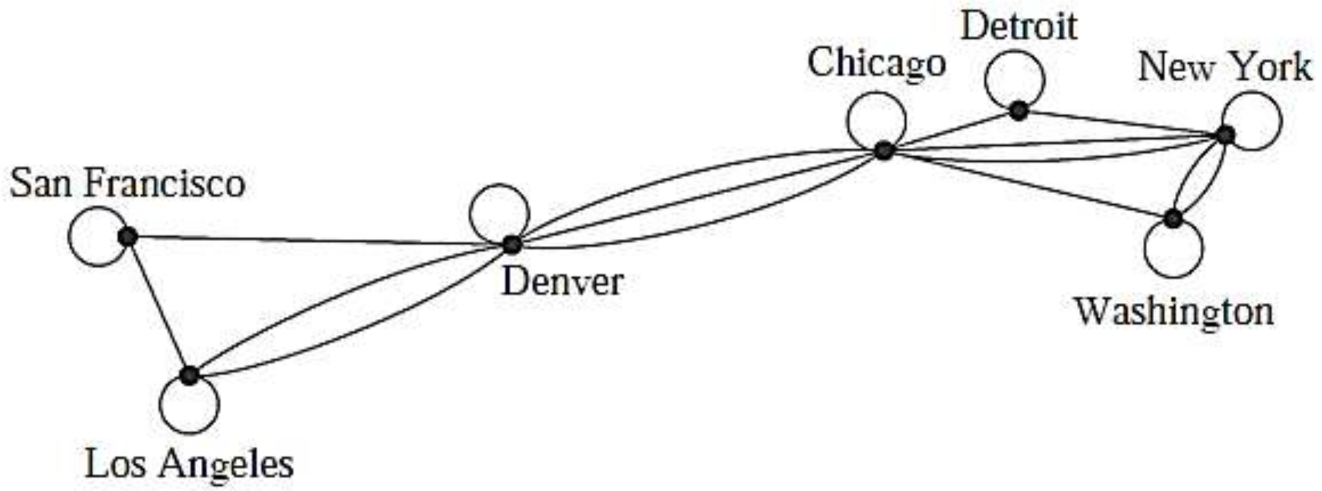
2. MULTIGRAPH

- A computer network may contain multiple links between data centers. To model such networks we need graphs that have more than one edge connecting the same pair of vertices.
- Graphs that may have **multiple edges** connecting the same vertices are called **multigraphs**.
- When there are ***m*** different edges associated to the same unordered pair of vertices we say that it is an edge of multiplicity *m*. That is, we can think of this set of edges as ***m*** different copies of an edge.
- They are **undirected graphs**. Their edges are also said to be **undirected. They don't contain any loop.**



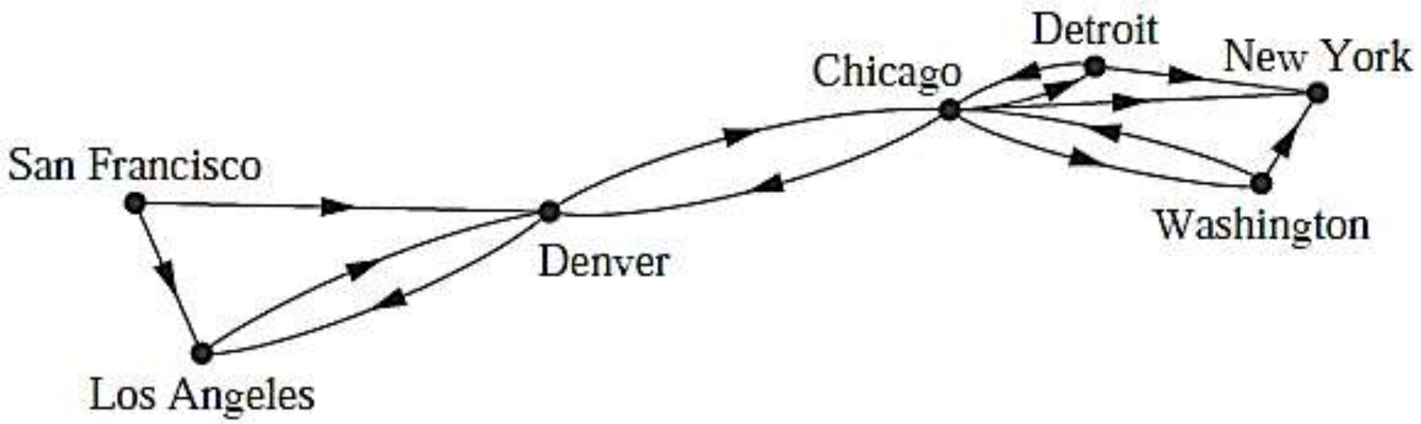
3. PSEUDOGRAPH

- Sometimes a communications link connects a data center with itself, perhaps a feedback loop for diagnostic purposes.
- To model this network we need to include edges that connect a vertex to itself. Such edges are called **loops**, and sometimes we may even have more than one loop at a vertex.
- Graphs that may include loops, and possibly multiple edges connecting the same pair of vertices or a vertex to itself, are sometimes called **pseudographs**.
- They are **undirected graphs**. Their edges are also said to be **undirected.**



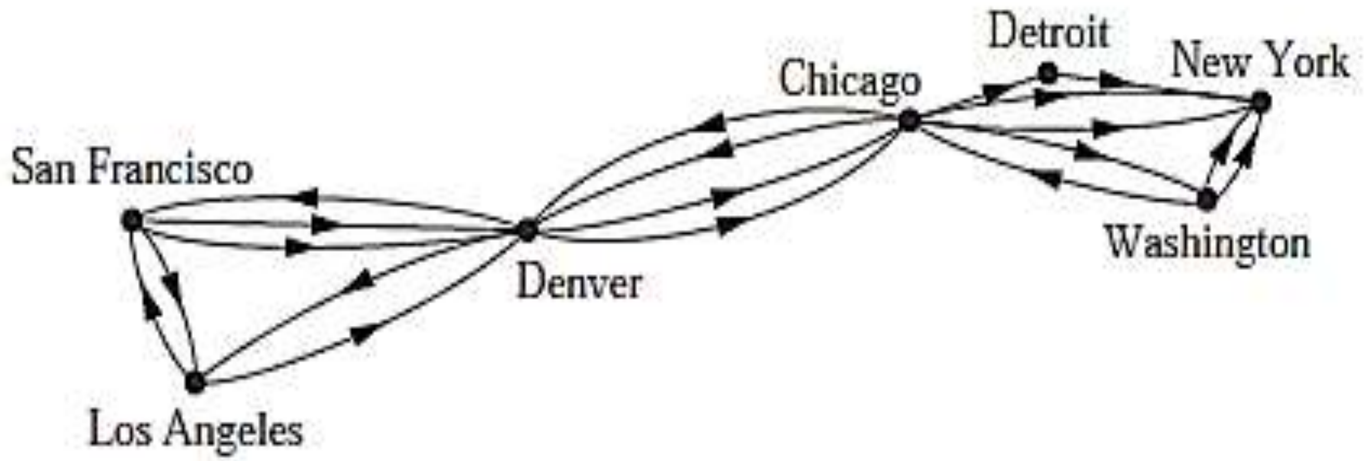
4. SIMPLE DIRECTED GRAPH

- To construct a graph model, we may find it necessary to assign directions to the edges of a graph. Each edge of a directed graph is associated to an ordered pair.
- When a directed graph has no loops and has no multiple directed edges, it is called a **simple directed graph**.
- Because a simple directed graph has at most one edge associated to each ordered pair of vertices *(u, v)*, we call *(u, v)* an edge if there is an edge associated to it in the graph.



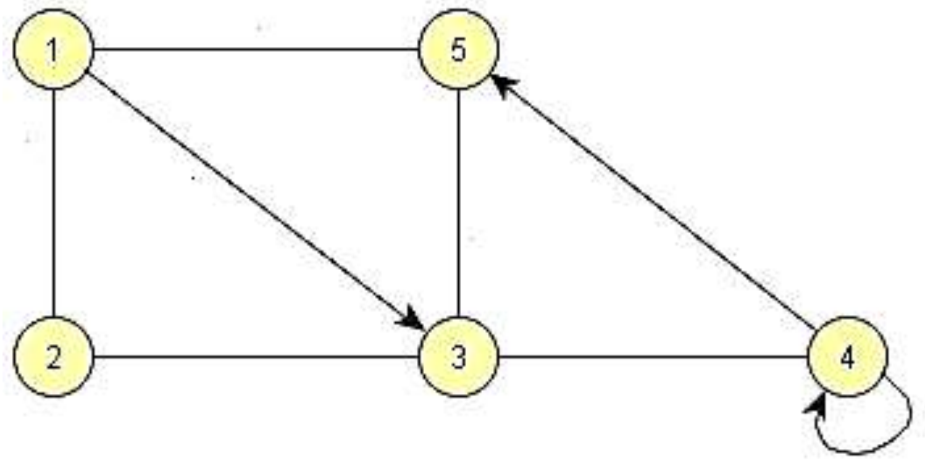
5. DIRECTED MULTIGRAPH

- Directed graphs that may have **multiple directed edges** from a vertex to a second (possibly the same) vertex are used to model such networks. We call such graphs **directed multigraphs**.
- When there are *m* directed edges, each associated to an ordered pair of vertices *(u, v)*, we say that *(u, v)* is an edge of **multiplicity** *m*



6. MIXED GRAPH

- A graph with both directed and undirected edges is called a **mixed graph**.
- For example, a mixed graph might be used to model a computer network containing links that operate in both directions and other links that operate only in one direction.



TYPES OF GRAPHS SUMMARIZED

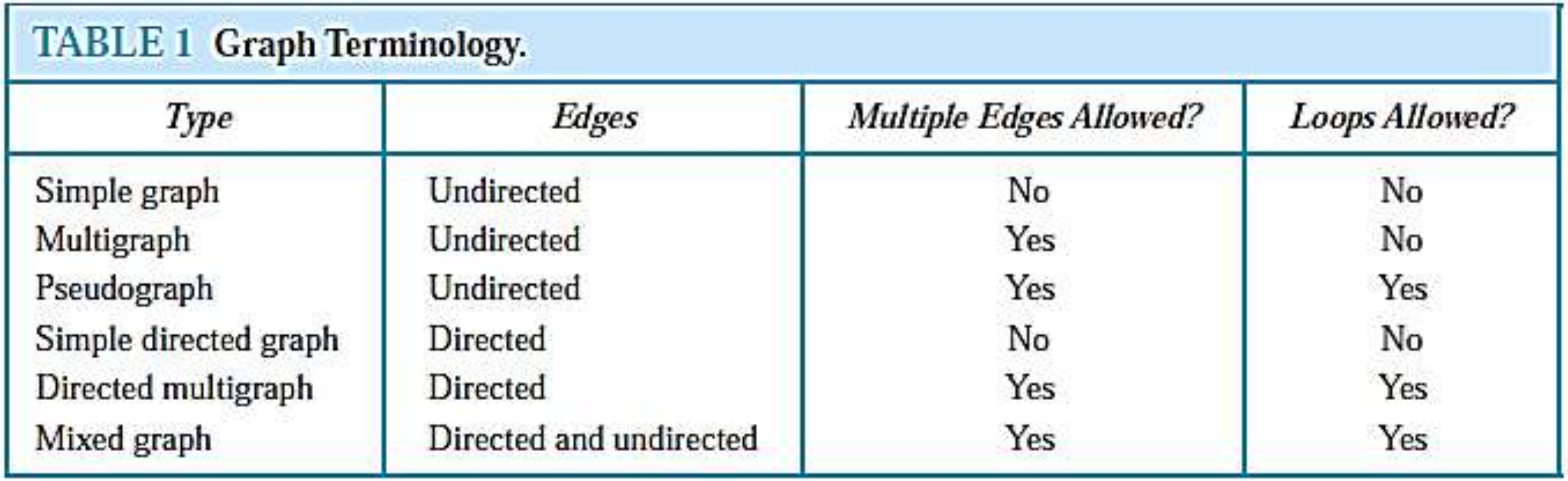


TABLE 1 Graph Terminology.

Type	*Edges*	*Multiple Edges Allowed?*	*Loops Allowed?*
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

STRUCTURE OF A GRAPH

Although the terminology used to describe graphs may vary, three key questions can help us understand the structure of a graph:

1. Are the edges of the graph undirected or directed (or both)?
2. If the graph is undirected, are multiple edges present that connect the same pair of vertices? If the graph is directed, are multiple directed edges present?
3. Are loops present?

Answering such questions helps us understand graphs.

GRAPH MODELS

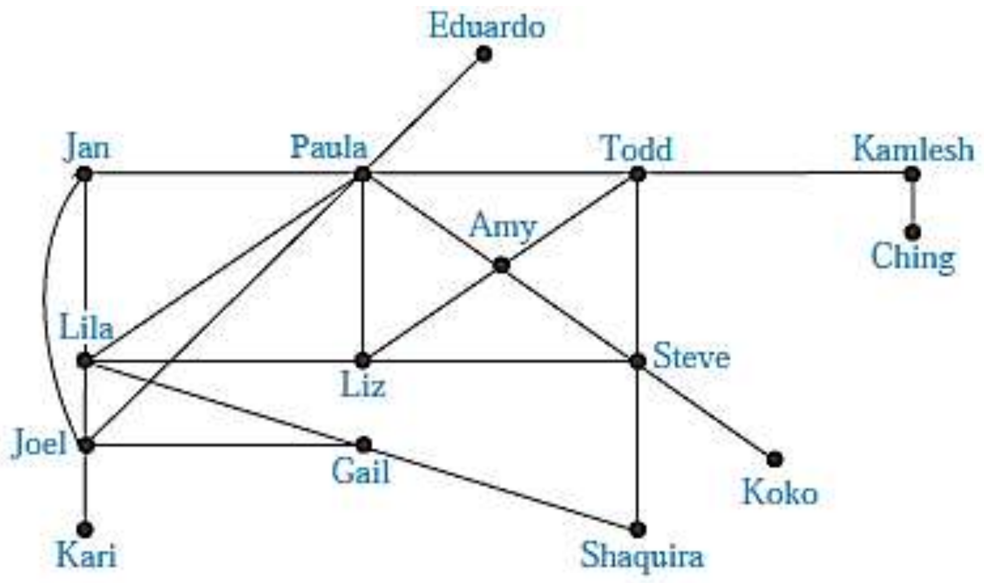
- Graphs are used in a wide variety of models.
- We began this section by describing how to construct graph models of communications networks linking data centers.
- We will complete this section by describing some diverse graph models for some interesting applications.

1. SOCIAL NETWORKS

- Graphs are extensively used to model social structures based on different kinds of **relationships between people or groups of people**.
- These social structures, and the graphs that represent them, are known as **social networks**. In these graph models, individuals or organizations are represented by vertices; relationships between individuals or organizations are represented by edges.
- The study of social networks is an extremely active multidisciplinary area, and many different types of relationships between people have been studied using them.

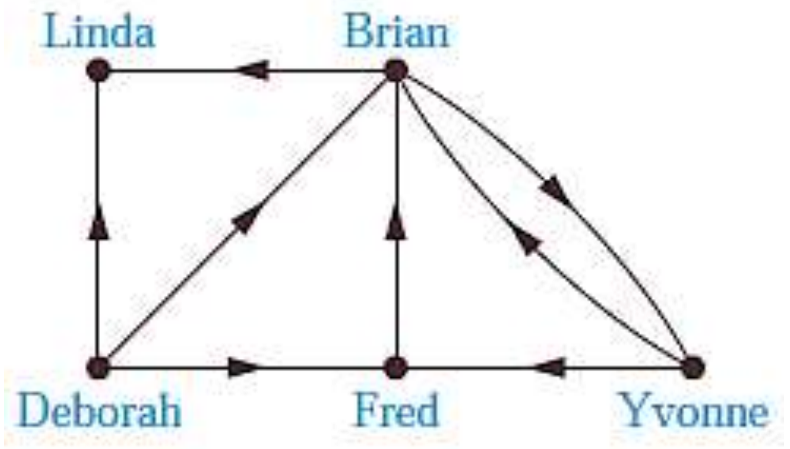
Example 1: Acquaintanceship and Friendship Graphs

- We can use a simple graph to represent whether two people know each other, that is, whether they are acquainted, or whether they are friends.
- Each person in a particular group of people is represented by a vertex. An **undirected** edge is used to connect two people when these people know each other.
- **No multiple edges and usually no loops are used.** (If we want to include the notion of self-knowledge, we would include loops.)



Example 2: Influence Graphs

- In studies of group behavior it is observed that certain people can influence the thinking of others. A directed graph called an **influence graph** can be used to model this behavior.
- Each person of the group is represented by a vertex.
- There is a **directed** edge from vertex *a* to vertex *b* when the person represented by vertex *a* can influence the person represented by vertex *b*.
- This graph **does not contain loops** and it **does not contain multiple directed edges.**



Example 3: Collaboration Graphs

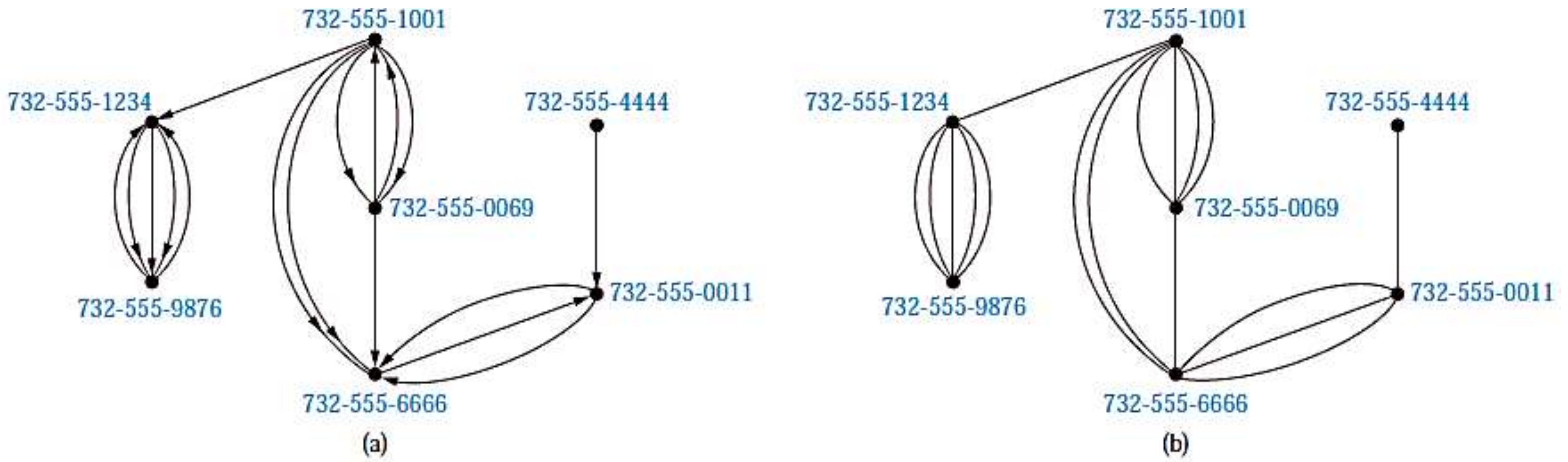
- A **collaboration graph** is used to model social networks where two people are related by working together in a particular way.
- Collaboration graphs are simple graphs, as edges in these graphs are **undirected** and there are **no multiple edges and no loops**.
- Vertices in these graphs represent people; two people are connected by an undirected edge when the people have collaborated.
- The **Hollywood graph** is a collaborator graph that represents actors by vertices and connects two actors with an edge if they have worked together on a movie or television show. The Hollywood graph is a huge graph with more than 1.5 million vertices (as of early 2011).
- In an **academic collaboration graph**, vertices represent people and edges link two people if they have jointly published a paper. The collaboration graph for people who have published research papers in mathematics was found in 2004 to have more than 400,000 vertices and 675,000 edges, and these numbers have grown considerably since then.
- Collaboration graphs have also been used in **sports**, where two professional athletes are considered to have collaborated if they have ever played on the same team during a regular season of their sport.

2. COMMUNICATION NETWORKS

- We can model different communications networks using vertices to represent devices and edges to represent the particular type of communications links of interest.
- **Example 4: Call Graphs**

- Graphs can be used to model telephone calls made in a network, such as a long distance telephone network.
- In particular, a **directed multigraph** can be used to model calls where each telephone number is represented by a vertex and each telephone call is represented by a directed edge. The edge representing a call starts at the telephone number from which the call was made and ends at the telephone number to which the call was made.
- We need directed edges because the direction in which the call is made matters. We need multiple directed edges because we want to represent each call made from a particular telephone number to a second number.

- The first graph shows, for instance, that three calls have been made from 732-555-1234 to 732-555-9876 and two in the other direction, but no calls have been made from 732-555-4444 to any of the other six numbers except 732-555-0011.
- When we care only whether there has been a call connecting two telephone numbers, we use an **undirected graph** with an edge connecting telephone numbers when there has been a call between these numbers. This version of the call graph is displayed in second graph.

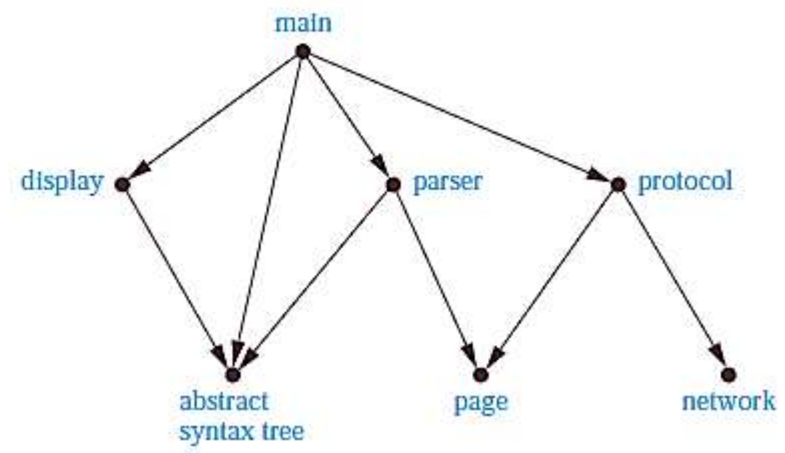


3. INFORMATION NETWORKS

- Graphs can be used to model various networks that link particular types of information. Here, we will describe how to model the WorldWideWeb using a graph. We will also describe how to use a graph to model the citations in different types of documents.
- **Example 5: The Web Graph**
- The WorldWideWeb can be modeled as a **directed graph** where each Web page is represented by a vertex and where an edge starts at the Web page *a* and ends at the Web page *b* if there is a link on *a* pointing to *b*. Because new Web pages are created and others removed somewhere on the Web almost every second, the Web graph changes on an almost continual basis.
- **Example 6: Citation Graphs**
- Graphs can be used to represent citations in different types of documents, including academic papers, patents, and legal opinions. In such graphs, each document is represented by a vertex, and there is an edge from one document to a second document if the first document cites the second in its citation list. (In an academic paper, the citation list is the bibliography, or list of references; in a patent it is the list of previous patents that are cited; and in a legal opinion it is the list of previous opinions cited.) A citation graph is a **directed graph without loops or multiple edges**.

4. SOFTWARE DESIGN APPLICATIONS

- Graph models are useful tools in the design of software. We will briefly describe two of these models here.
- **Example 7: Module Dependency Graphs**
- One of the most important tasks in designing software is how to structure a program into different parts, or modules.



•Understanding how the different modules of a program interact is essential not only for program design, but also for testing and maintenance of the resulting software.

•A **module dependency graph** provides a useful tool for understanding how different modules of a program interact. In a program dependency graph, each module is represented by a vertex. There is a **directed edge from a module to a second module if the second module depends on the first.**

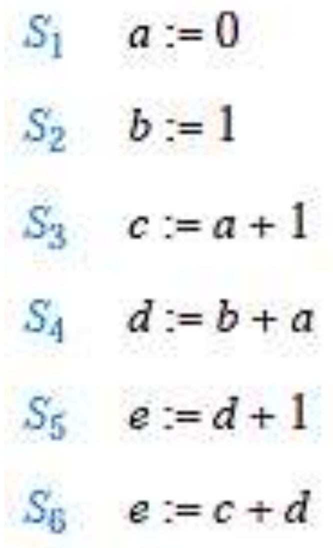
Example 8: Precedence Graphs and Concurrent Processing

- Computer programs can be executed more rapidly by executing certain statements concurrently. It is important not to execute a statement that requires results of statements not yet executed.
- The dependence of statements on previous statements can be represented by a **directed graph**. Each statement is represented by a vertex, and there is an edge from one statement to a second statement if the second statement cannot be executed before the first statement. This resulting graph is called a **precedence graph**.
- A computer program and its graph are displayed in

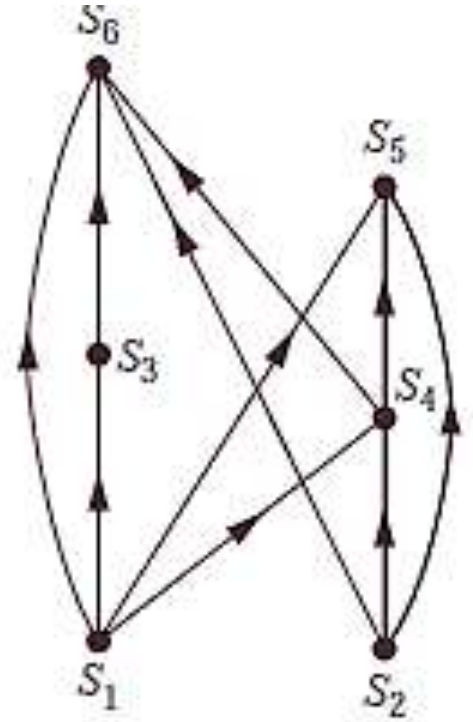
Figure. For instance, the graph shows that statement

*S*5 cannot be executed before statements *S*1, *S*2,

and *S*4 are executed.



S_1	$a := 0$
S_2	$b := 1$
S_3	$c := a + 1$
S_4	$d := b + a$
S_5	$e := d + 1$
S_6	$e := c + d$



5. TRANSPORTATION NETWORKS

- We can use graphs to model many different types of transportation networks, including road, air, and rail networks, as well shipping networks.
- **Example 9: Airline Routes**

- We can model airline networks by representing each airport by a vertex.
- In particular, we can model all the flights by a particular airline each day using a directed edge to represent each flight, going from the vertex representing the departure airport to the vertex representing the destination airport. The resulting graph will generally be a **directed multigraph**, as there may be multiple flights from one airport to some other airport during the same day.

BASIC GRAPH TERMINOLOGIES

- Two vertices *u* and *v* in an undirected graph *G* are called ***adjacent*** **(or *neighbors*)** in *G* if *u* and *v* are endpoints of an edge *e* of *G*. Such an edge *e* is called ***incident*** *with* the vertices *u* and *v* and *e* is said to ***connect*** *u* and *v*.

- The set of all neighbors of a vertex *v* of *G = (V ,E)*, denoted by *N(v)*, is called the ***neighborhood*** of *v*. If *A* is a subset of *V*, we denote by *N(A)* the set of all vertices in *G* that are adjacent to at least one vertex in *A*.

- The ***degree of a vertex*** *in an undirected graph* is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex. The degree of the vertex *v* is denoted by deg*(v)*.

- A vertex of degree zero is called **isolated**. It follows that an isolated vertex is not adjacent to any vertex.

- vertex is **pendant** if and only if it has degree one.

EXAMPLE 10

For the graph shown below

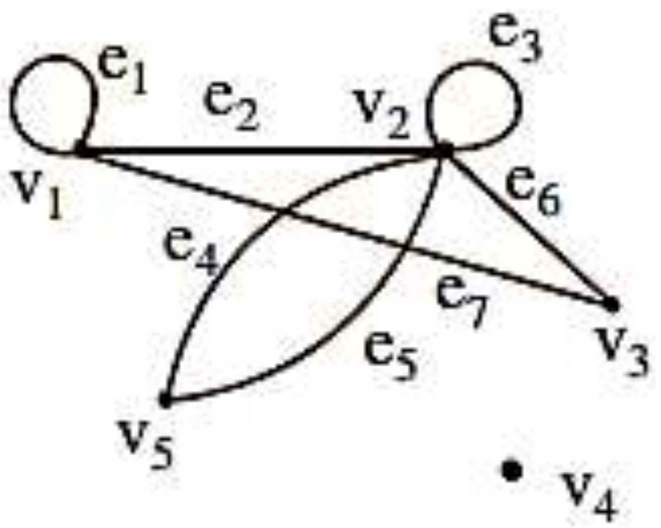
(i) find all edges that are incident on v_1;

(ii)find all vertices that are adjacent to v_3;

(iii)find all loops;

(iv)find all parallel edges;

(v)find all isolated vertice



EXAMPLE 10

For the graph shown below

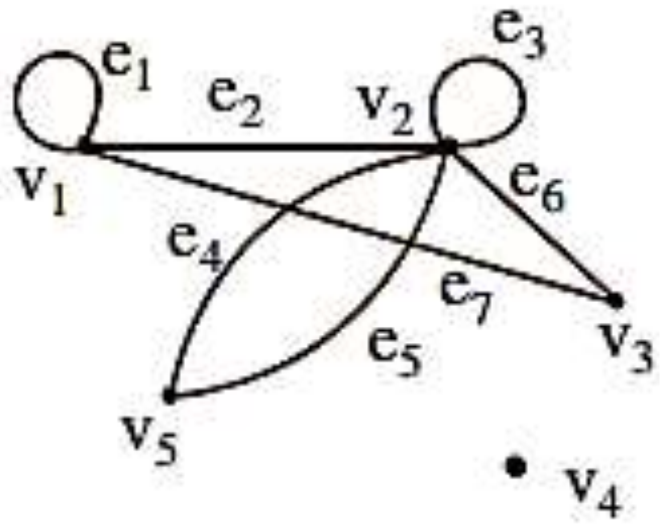
(i) find all edges that are incident on v_1;

(ii)find all vertices that are adjacent to v_3;

(iii)find all loops;

(iv)find all parallel edges;

(v)find all isolated vertices;



SOLUTION:

(i) v_1 is incident with edges e_1, e_2 and e_7

(ii) vertices adjacent to v_3 are v_1 and v_2

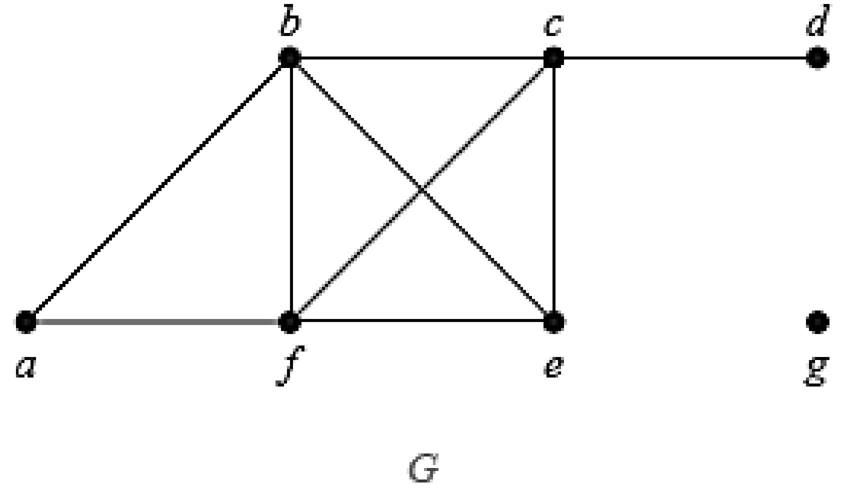
(iii) loops are e_1 and e_3

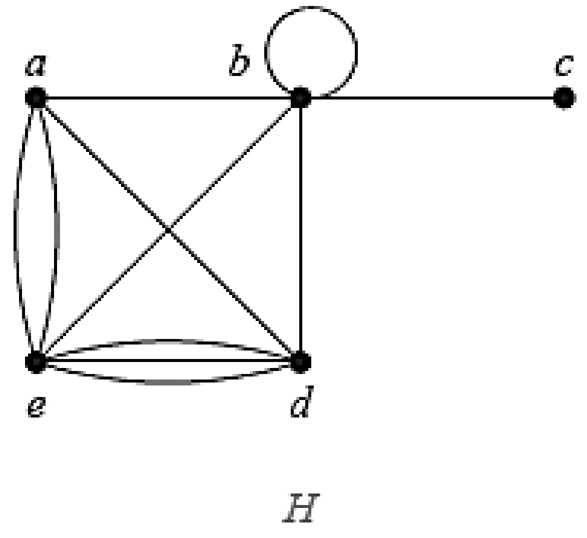
(iv) only edges e_4 and e_5 are parallel

(v) The only isolated vertex is v_4 in this Graph.

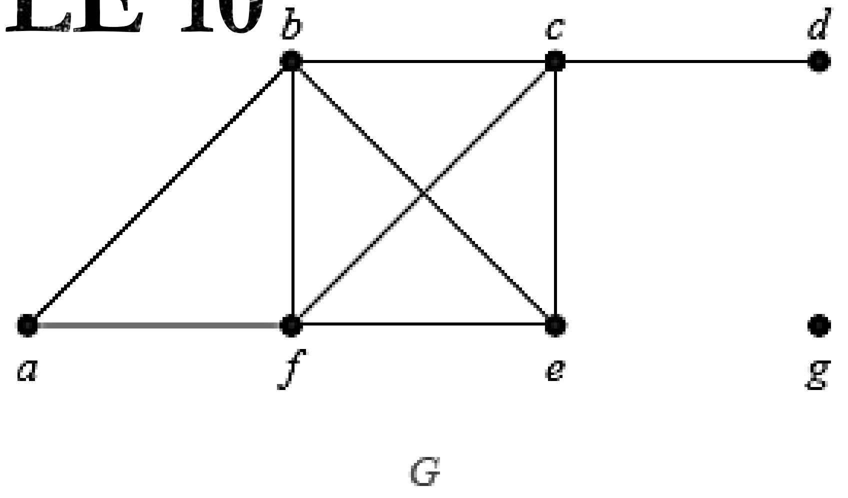
EXAMPLE 10

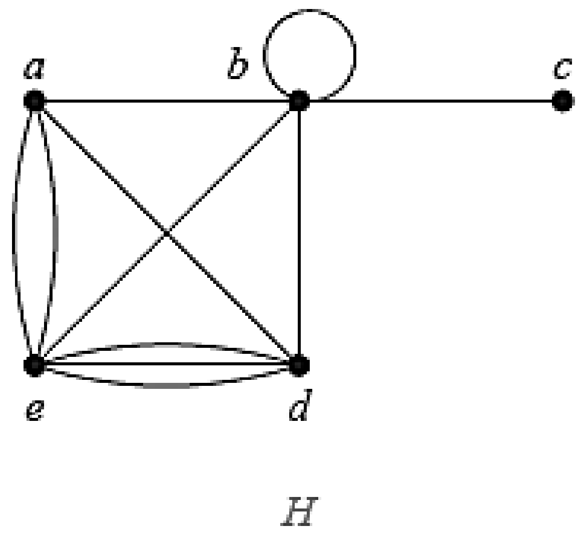
What are the degrees and what are the neighborhoods of the vertices in the graphs G and H shown below? Also find the degree of G and H.





EXAMPLE 10





- **In G,**
  - deg(a) = 2, deg(b) = deg(c) = deg(f) = 4, deg(d) = 1, deg(e) = 3, and deg(g) = 0.
  - The neighborhoods of these vertices are N(a) = {b, f }, N(b) = {a, c, e, f }, N(c) = {b, d, e, f }, N(d) = {c}, N(e) = {b, c, f }, N(f) = {a, b, c, e}, and N(g) = ∅.
  - deg(G) = 18.

- **In H,**
  - deg(a) = 4, deg(b) = deg(e) = 6, deg(c) = 1, and deg(d) = 5.
  - The neighborhoods of these vertices are N(a) = {b, d, e}, N(b) = {a, b, c, d, e}, N(c) = {b}, N(d) = {a, b, e}, and N(e) = {a, b, d}.
- deg(H) = 22.

DEGREE OF A VERTEX:

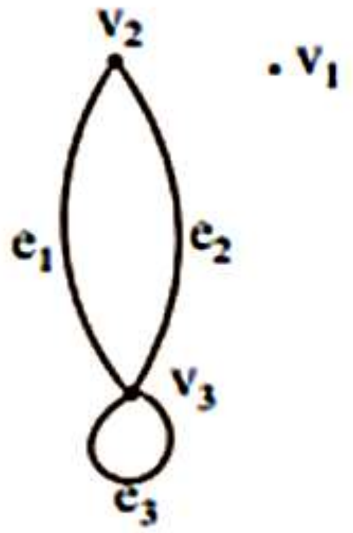
Let G be a graph and v a vertex of G. The degree of v, denoted **deg(v),** equals the number of edges that are incident on v, with an edge that is a loop counted twice.

Note:(i)The total degree of G is the sum of the degrees of all the vertices of G.

(ii) The **degree** of a loop is counted twice.

EXAMPLE:11

For the graph shown



deg (v_1) = 0, since v_1 is isolated vertex.
deg (v_2) = 2, since v_2 is incident on e_1 and e_2 .
deg (v_3) = 4, since v_3 is incident on e_1,e_2 and the loop e_3.
Total degree of G = deg(v_1) + deg(v_2) + deg(v_3)
= 0 + 2 + 4
= 6

REMARK:

The total degree of G, which is 6, equals twice the number of edges of G, which is 3.

THE HANDSHAKING THEOREM

Each edge contributes two to the sum of the degrees of the vertices because an edge is incident with exactly two (possibly equal) vertices. **This means that the sum of the degrees of the vertices is twice the number of edges.**

THE HANDSHAKING THEOREM Let $G = (V, E)$ be an undirected graph with m edges. Then

$$2m = \sum_{v \in V} \deg(v).$$

i.e. The total degree of G = 2 x (the number of edges of G)

(Note that this applies even if multiple edges and loops are present.)

EXAMPLE 12

- How many edges are there in a graph with 10 vertices each of degree six?

EXAMPLE 12

- How many edges are there in a graph with 10 vertices each of degree six?
- *Solution:*
- Because the sum of the degrees of the vertices is $6 \cdot 10 = 60$, it follows that $2m = 60$ where m is the number of edges.
- Therefore, $m = 30$.

EXAMPLE 13

Suppose a graph has vertices of degrees 1, 1, 4, 4 and 6. How many edges does the graph have?

SOLUTION:

The total degree of graph = 1 + 1 + 4 + 4 + 6

= 16

16 = 2.(number of edges of graph) [by using Handshaking theorem]

Number of edges of graph = $\frac{16}{2} = 8$

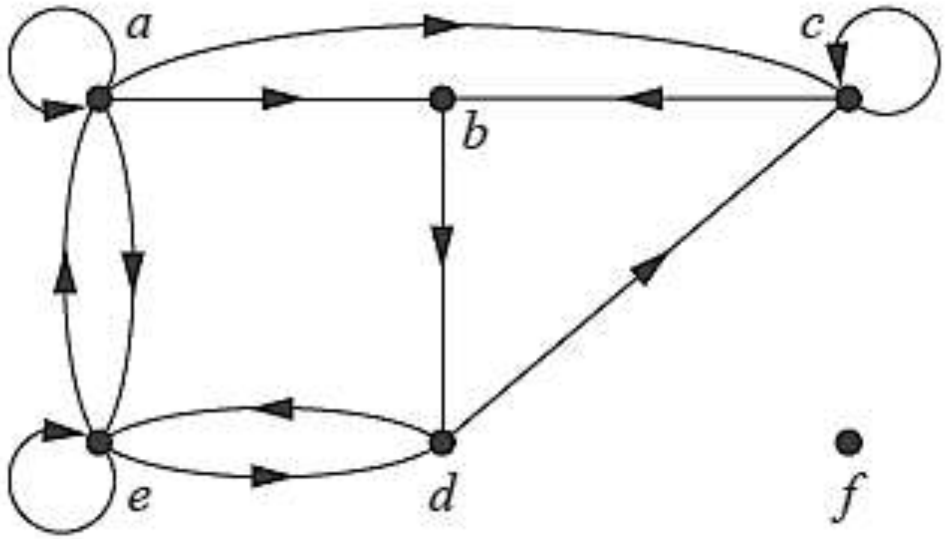
IN DEGREE OUT DEGREE

When (u, v) is an edge of the graph G with directed edges, u is said to be *adjacent to* v and v is said to be *adjacent from* u. The vertex u is called the *initial vertex* of (u, v), and v is called the *terminal* or *end vertex* of (u, v). The initial vertex and terminal vertex of a loop are the same.

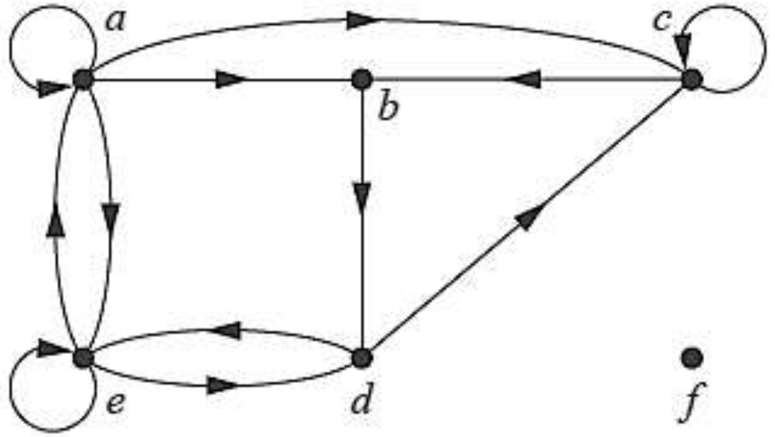
In a graph with directed edges the *in-degree of a vertex* v, denoted by $\deg^-(v)$, is the number of edges with v as their terminal vertex. The *out-degree of* v, denoted by $\deg^+(v)$, is the number of edges with v as their initial vertex. (Note that a loop at a vertex contributes 1 to both the in-degree and the out-degree of this vertex.)

EXAMPLE 14

- Find the in-degree and out-degree of each vertex in the graph *G* with directed edges shown below.



EXAMPLE 14



Solution: The in-degrees in G are $\deg^-(a) = 2$, $\deg^-(b) = 2$, $\deg^-(c) = 3$, $\deg^-(d) = 2$, $\deg^-(e) = 3$, and $\deg^-(f) = 0$. The out-degrees are $\deg^+(a) = 4$, $\deg^+(b) = 1$, $\deg^+(c) = 2$, $\deg^+(d) = 2$, $\deg^+(e) = 3$, and $\deg^+(f) = 0$. ◀

Because each edge has an initial vertex and a terminal vertex, the sum of the in-degrees and the sum of the out-degrees of all vertices in a graph with directed edges are the same. Both of these sums are the number of edges in the graph. This result is stated as

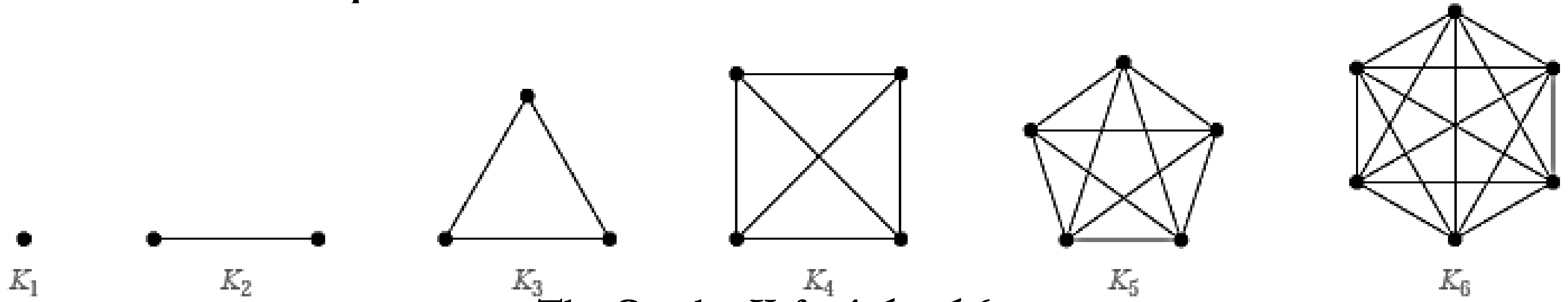
Let $G = (V, E)$ be a graph with directed edges. Then

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|.$$

SOME SPECIAL SIMPLE GRAPHS

- **Complete Graphs**

A **complete graph on *n* vertices**, denoted by K_n, is a simple graph that contains exactly one edge between each pair of distinct vertices. The graphs K_n, for $n = 1, 2, 3, 4, 5, 6$, are displayed in Figure. In short, **Every vertex is connected to every other vertex**. A simple graph for which there is at least one pair of distinct vertex not connected by an edge is called **non-complete**.

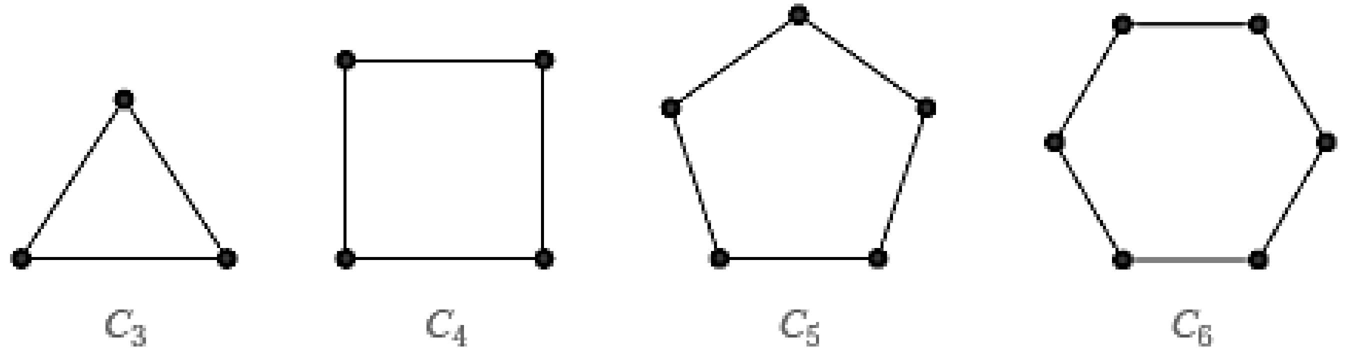


The Graphs K_n for $1 \leq n \leq 6$.

SOME SPECIAL SIMPLE GRAPHS

- **Cycles**

A **cycle** $\boldsymbol{C_n}$, $n \geq 3$, consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}$, $\{v_2, v_3\}$,..., $\{v_{n-1}, v_n\}$, and $\{v_n, v_1\}$. The cycles C_3, C_4, C_5, and C_6 are displayed in Figure.

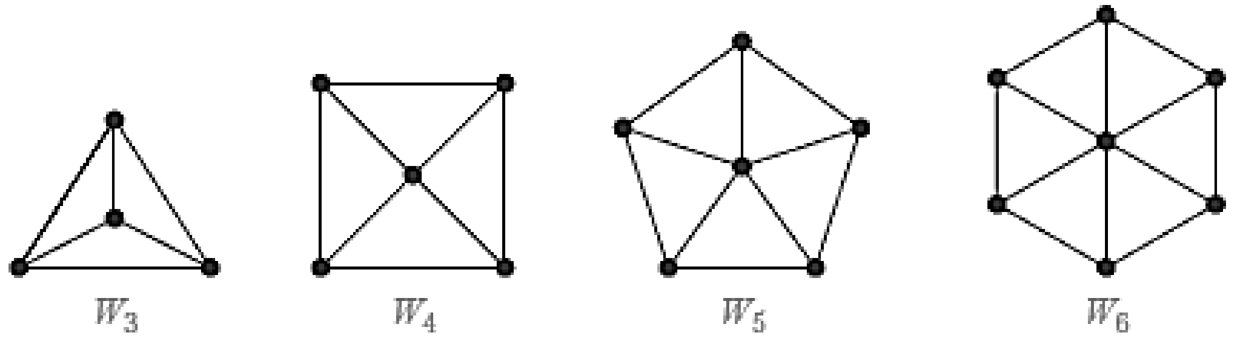


The Cycles C_3, C_4, C_5, and C_6.

SOME SPECIAL SIMPLE GRAPHS

- **Wheels**

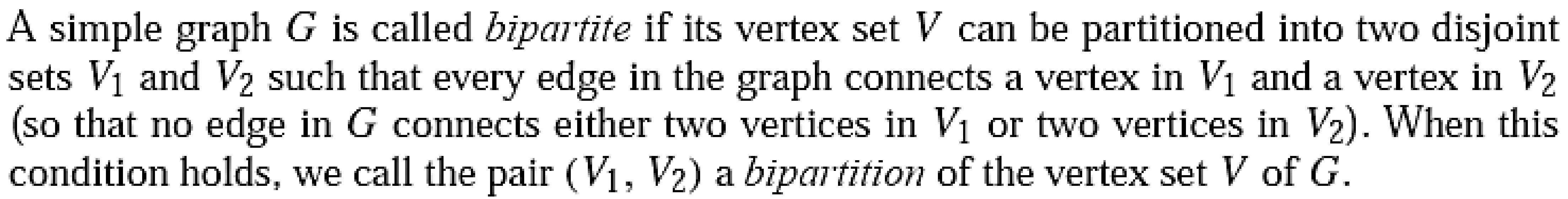
We obtain a **wheel** $\boldsymbol{W_n}$ when we add an additional vertex to a cycle C_n, for $n \geq 3$, and connect this new vertex to each of the n vertices in C_n, by new edges. The wheels W_3, W_4, W_5, and W_6 are displayed in Figure.



The Wheels W_3, W_4, W_5, and W_6

BIPARTITE GRAPHS

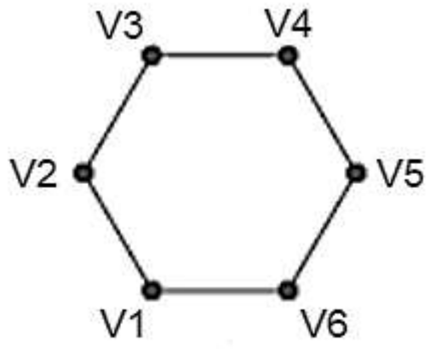
- Sometimes a graph has the property that its vertex set can be divided into two disjoint subsets such that each edge connects a vertex in one of these subsets to a vertex in the other subset.



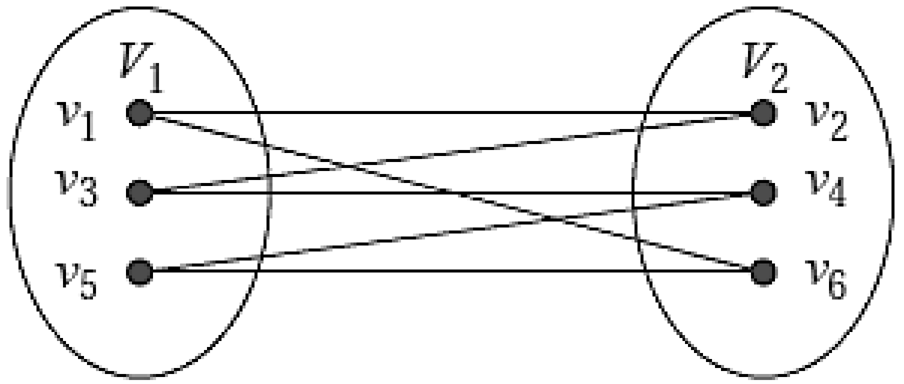
A simple graph G is called *bipartite* if its vertex set V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2). When this condition holds, we call the pair (V_1, V_2) a *bipartition* of the vertex set V of G.

EXAMPLE 15

- Is C_6 a bipartite?

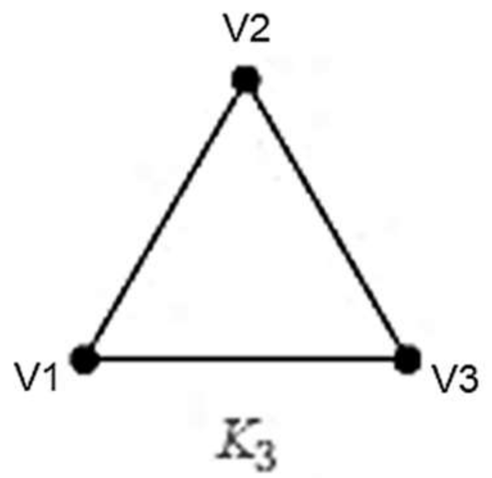


Solution:
C_6 is bipartite, because its vertex set can be partitioned into the two sets $V1=\{v1, v3, v5\}$ and $V2=\{v2, v4, v6\}$, and every edge of C_6 connects a vertex in $V1$ and a vertex in $V2$.



EXAMPLE 16

- Is K_3 a bipartite?



Solution:

K_3 is not bipartite. To verify this, note that if we divide the vertex set of K_3 into two disjoint sets, one of the two sets must contain two vertices. If the graph were bipartite, these two vertices could not be connected by an edge, but in K_3 each vertex is connected to every other vertex by an edge.

How to Find if a Graph is Bipartite or not?

1. Path Finding

A graph is bipartite if and only if it is not possible to start at a vertex and return to this vertex by traversing an odd number of distinct edges.

2. GRAPH COLORINGS

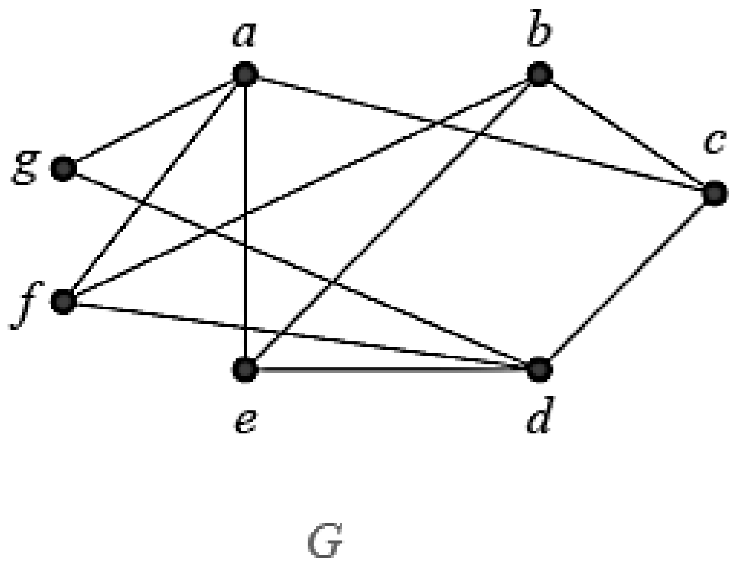
The coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

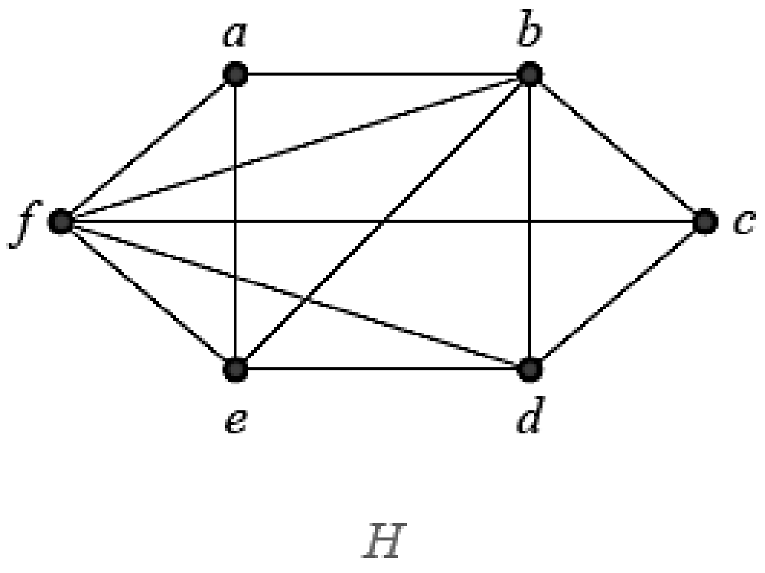
Theorem:

A simple graph is bipartite if and only if it is possible to assign one of two different colors to each vertex of the graph so that no two adjacent vertices are assigned the same color.

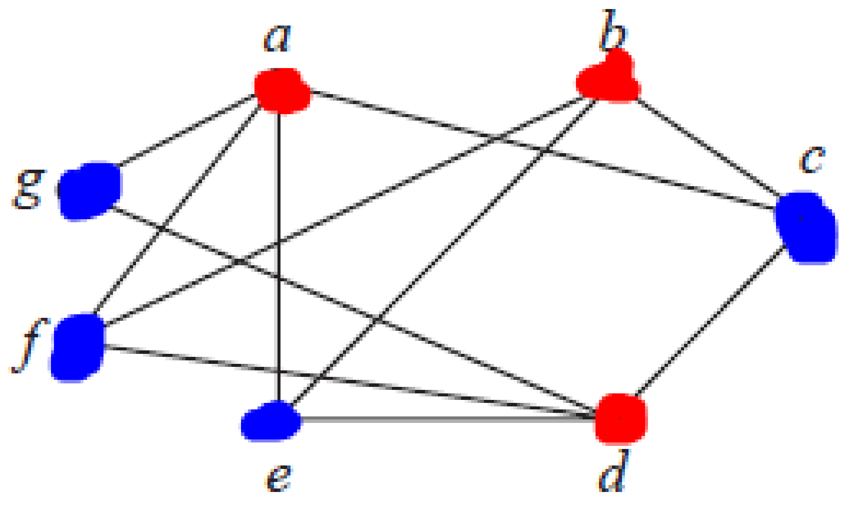
EXAMPLE 17

- Are the graphs *G* and *H* displayed in Figure bipartite (Hint: use the theorem above?





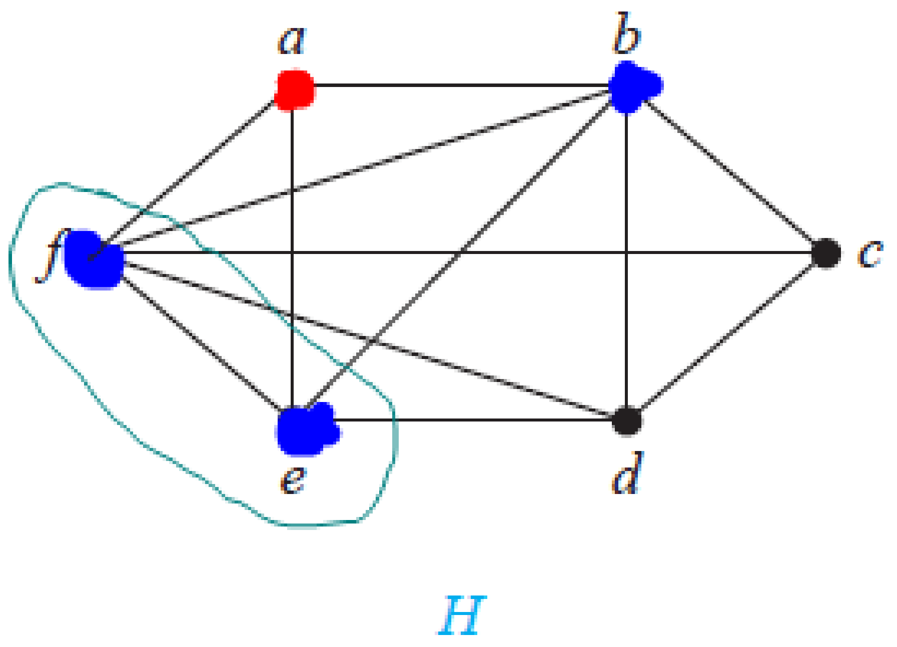
EXAMPLE 17



G

- Consider the graph *G*. We will try to assign one of two colors, say red and blue, to each vertex in *G* so that no edge in *G* connects a red vertex and a blue vertex.
- Without loss of generality we begin by arbitrarily assigning red to *a*. Then, we must assign blue to *c*, *e*, *f* , and *g*, because each of these vertices is adjacent to *a*. To avoid having an edge with two blue endpoints, we must assign red to all the vertices adjacent to either *c*, *e*, *f* , or *g*.
- This means thatwe must assign red to both *b* and *d* (and means that *a* must be assigned red, which it already has been).
- We have now assigned colors to all vertices, with *a*, *b*, and *d* red and *c*, *e*, *f* , and *g* blue.
- Checking all edges, we see that every edge connects a red vertex and a blue vertex.
- **Hence, by theorem the graph *G* is bipartite.**

EXAMPLE 17



- Next, we will try to assign either red or blue to each vertex in *H* so that no edge in *H* connects a red vertex and a blue vertex.
- Without loss of generality we arbitrarily assign red to *a*. Then, we must assign blue to *b*, *e*, and *f* , because each is adjacent to *a*.
- But this is not possible because *e* and *f* are adjacent, so both cannot be assigned blue.
- This argument shows that we cannot assign one of two colors to each of the vertices of *H* so that no adjacent vertices are assigned the same color.
- **It follows by Theorem *H* is not bipartite.**

REPRESENTING GRAPHS

A graph can be represented by any of the following method:

1. Adjacency List
2. Adjacency Matrix
3. Incidence Matrix

REPRESENTING GRAPHS: 1. ADJACENCY LISTS

- One way to represent a graph with no multiple edges is to use adjacency lists, which specify the vertices that are adjacent to each vertex of the graph.

- **Example 18:**

Use adjacency lists to describe the simple graph given in Figure.

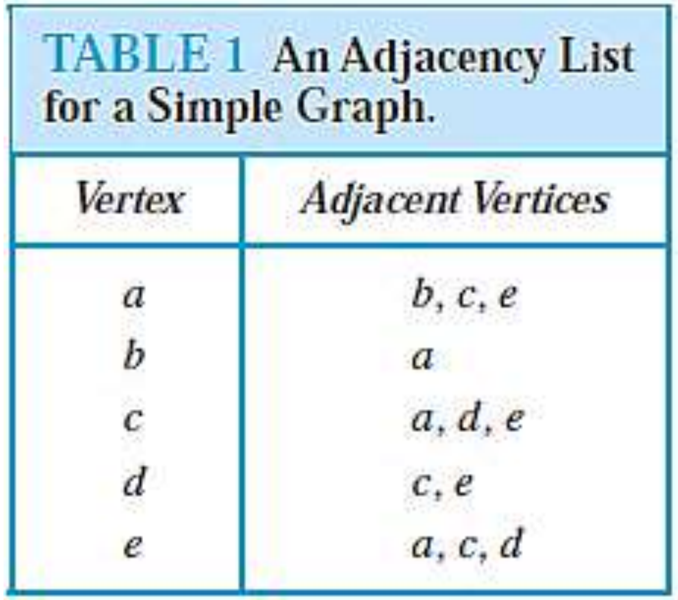
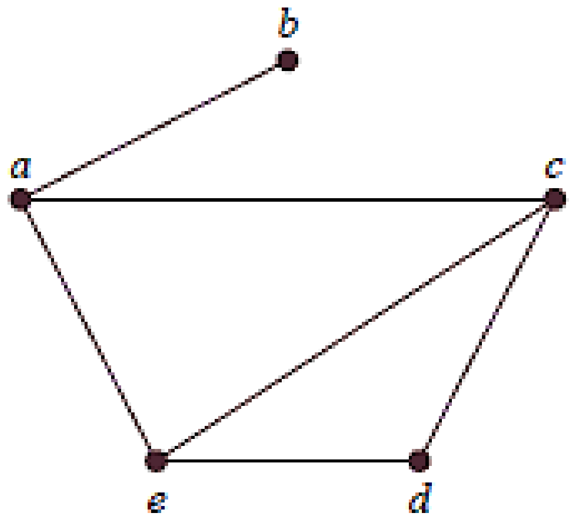


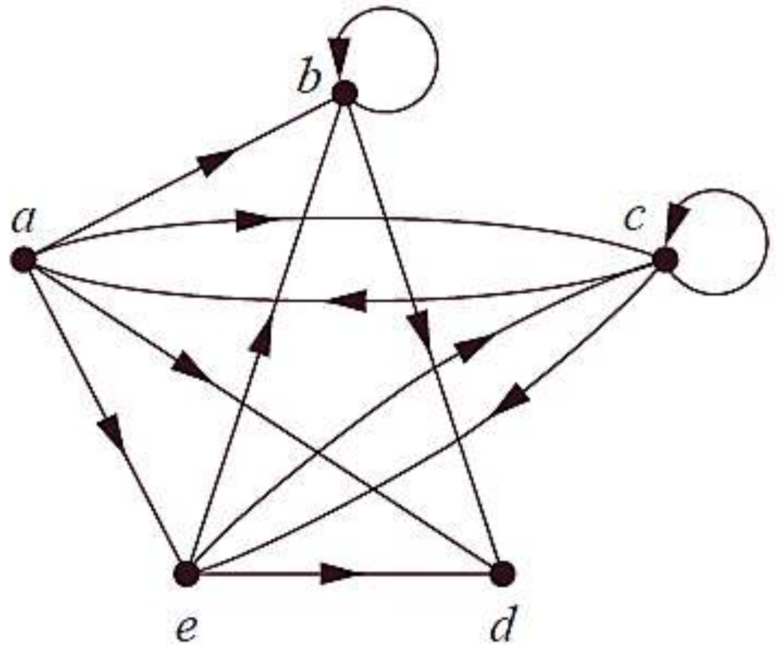
TABLE 1 An Adjacency List for a Simple Graph.

Vertex	*Adjacent Vertices*
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d



EXAMPLE 19

- Represent the directed graph shown in Figure below by listing all the vertices that are the terminal vertices of edges starting at each vertex of the graph.



EXAMPLE 19

- Represent the directed graph shown in Figure below by listing all the vertices that are the terminal vertices of edges starting at each vertex of the graph.

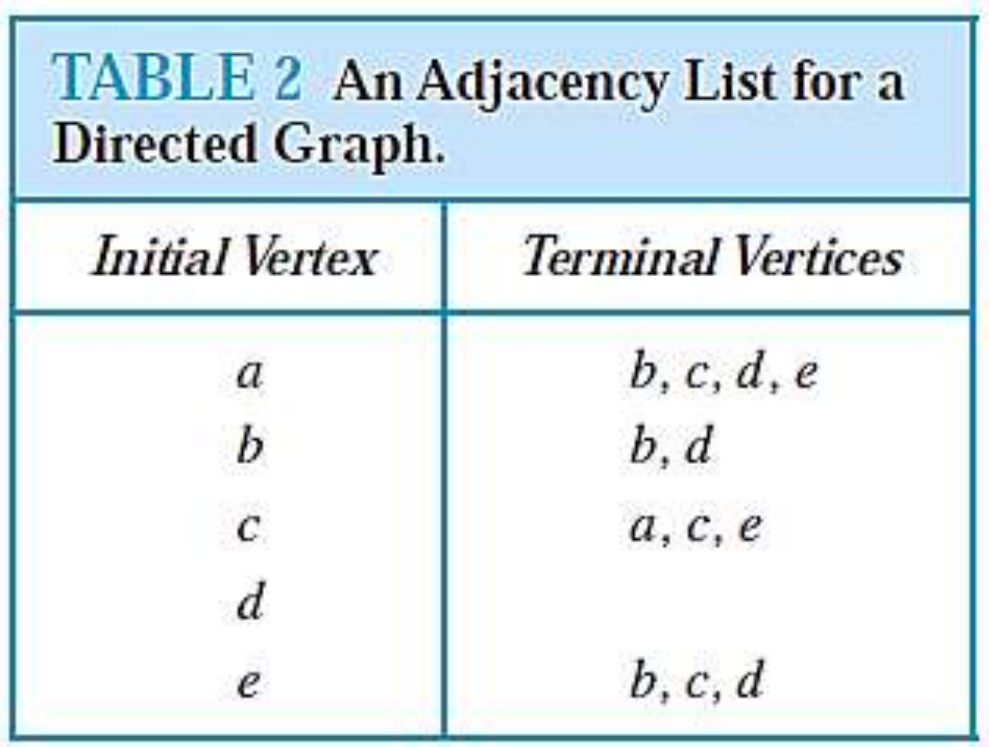
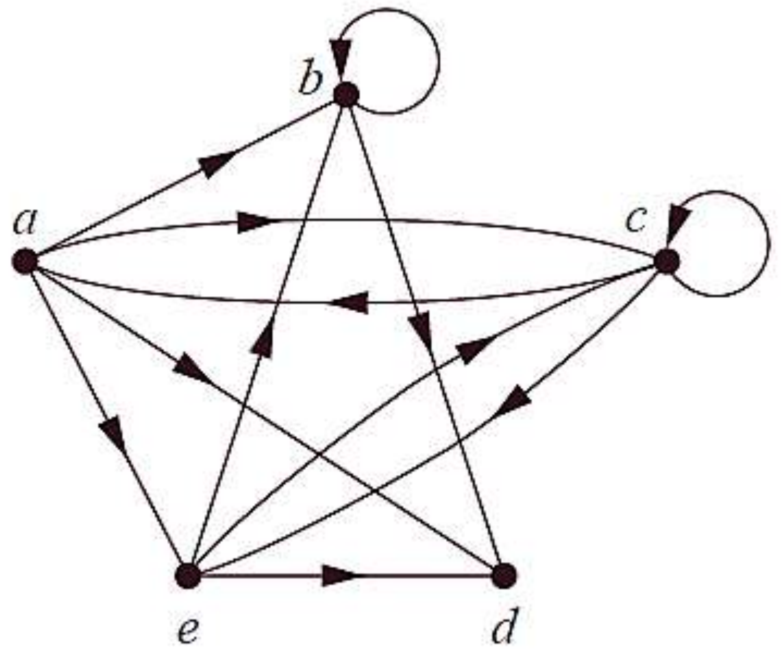


TABLE 2 An Adjacency List for a Directed Graph.

Initial Vertex	*Terminal Vertices*
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d



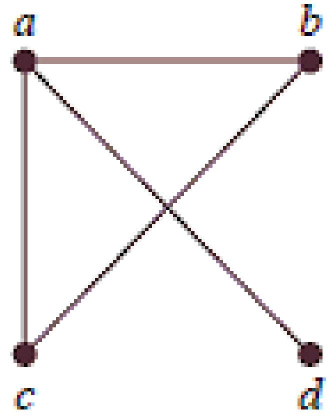
REPRESENTING GRAPHS: 2. ADJACENCY MATRICES

Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. Suppose that the vertices of G are listed arbitrarily as $v_1, v_2, \ldots, v_n$. The **adjacency matrix** A (or A_G) of G, with respect to this listing of the vertices, is the n x n zero–one matrix with 1 as its (i, j)th entry when v_i and v_j are adjacent, and 0 as its (i, j)th entry when they are not adjacent. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

EXAMPLE 20

- Use an adjacency matrix to represent the graph shown in Figure.



- Solution:

We order the vertices as *a*, *b*, *c*, *d*. The matrix representing this graph is

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

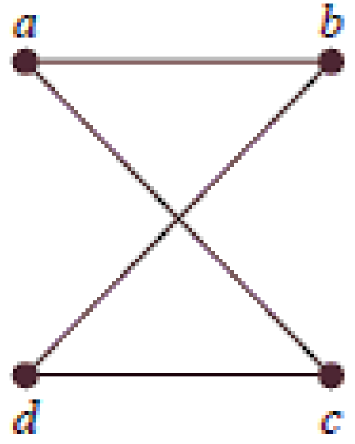
EXAMPLE 21

- Draw a graph with the adjacency matrix

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

- Solution:

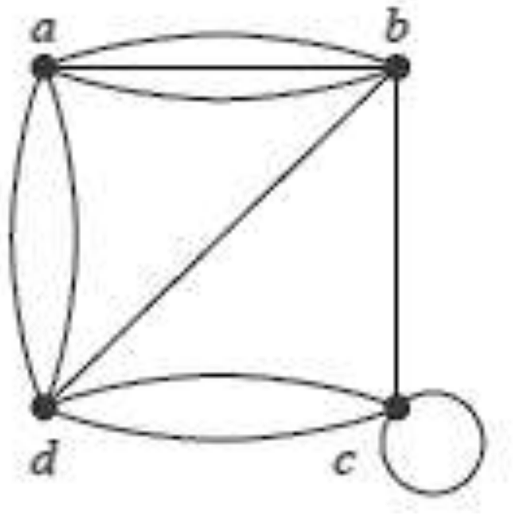
The graph with this adjacency matrix is shown in Figure



- **Note that an adjacency matrix of a graph is based on the ordering chosen for the vertices.** Hence, there may be as many as *n*! different adjacency matrices for a graph with *n* vertices, because there are *n*! different orderings of *n* vertices.
- **The adjacency matrix of a simple graph is symmetric**, that is, *aij* = *aji* , because both of these entries are 1 when *vi* and *vj* are adjacent, and both are 0 otherwise.
- Furthermore, **because a simple graph has no loops, each entry *aii, i* = 1, 2, 3, . . . , *n*, is 0.**
- **Adjacency matrices can also be used to represent undirected graphs with loops and with multiple edges.** A loop at the vertex *vi* is represented by a 1 at the *(i, i)*th position of the adjacency matrix. When multiple edges connecting the same pair of vertices *vi* and *vj*, or multiple loops at the same vertex, are present, the adjacency matrix is no longer a zero–one matrix, because the *(i, j)*th entry of this matrix equals the number of edges that are associated to {*vi* , *vj*}.
- **All undirected graphs, including multigraphs and pseudographs, have symmetric adjacency matrices.**

EXAMPLE 22

- Use an adjacency matrix to represent the pseudograph shown in Figure.



- **Solution:**

The adjacency matrix using the ordering of vertices *a, b, c, d* is

$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

REPRESENTING GRAPHS: 3. INCIDENCE MATRICES

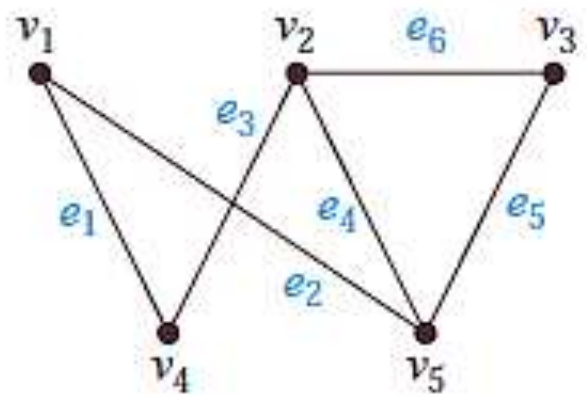
Another common way to represent graphs is to use **incidence matrices**. Let $G = (V, E)$ be an undirected graph. Suppose that $v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of G. Then the incidence matrix with respect to this ordering of V and E is the $n \times m$ matrix $\mathbf{M} = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

- **Incidence matrices can also be used to represent multiple edges and loops.**
- Multiple edges are represented in the incidence matrix using columns with identical entries, because these edges are incident with the same pair of vertices.
- Loops are represented using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with this loop.

EXAMPLE 23

- Represent the graph shown in Figure with an incidence matrix.



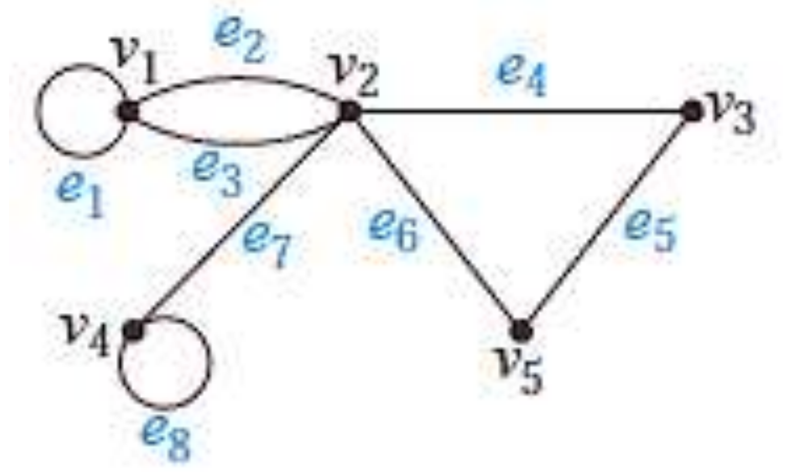
- **Solution:**

The incidence matrix is

$$\begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \hline v_1 & 1 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 0 & 0 & 1 & 1 & 0 & 1 \\ v_3 & 0 & 0 & 0 & 0 & 1 & 1 \\ v_4 & 1 & 0 & 1 & 0 & 0 & 0 \\ v_5 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

EXAMPLE 24

- Represent the pseudograph shown in Figure using an incidence matrix.



- **Solution:**

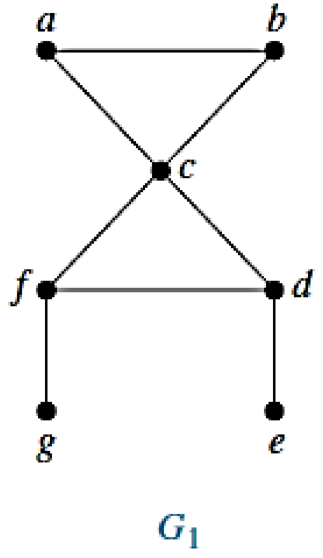
The incidence matrix for this graph is

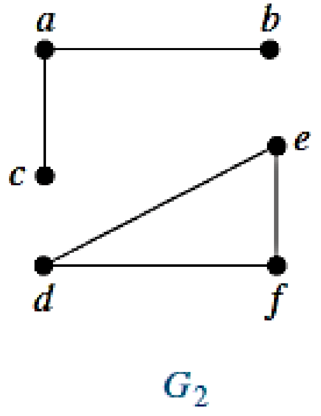
$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}\begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \left[\begin{array}{cccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right] \end{array}$$

CONNECTED GRAPH

- An undirected graph is called **connected** if there is a path between every pair of distinct vertices of the graph.

- The graph ***G*1** is connected.
- The graph ***G*2** is not connected. (For instance, there is no path in *G*2 between vertices *a* and *d*.)





GRAPH CONNECTIVITY

Let G be a graph, and let v and w be vertices in G.
A **walk** from v to w is a finite alternating sequence of adjacent vertices and edges of G. Thus a walk has the form

$$v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n,$$

where the v's represent vertices, the e's represent edges, $v_0 = v$, $v_n = w$, and for all i = 1; 2;…. n, v_{i-1} and v_i are the endpoints of e_i .

A **path** from v to w is a trail that does not contain a repeated vertex.

GRAPH CONNECTIVITY

- A **Closed walk** is a walk that starts and ends at the same vertex.
- A **circuit** is a closed walk that contains at least one edge and does not contain a repeated edge.
- A **simple circuit** is a circuit that does not have any other repeated vertex except the first and last.

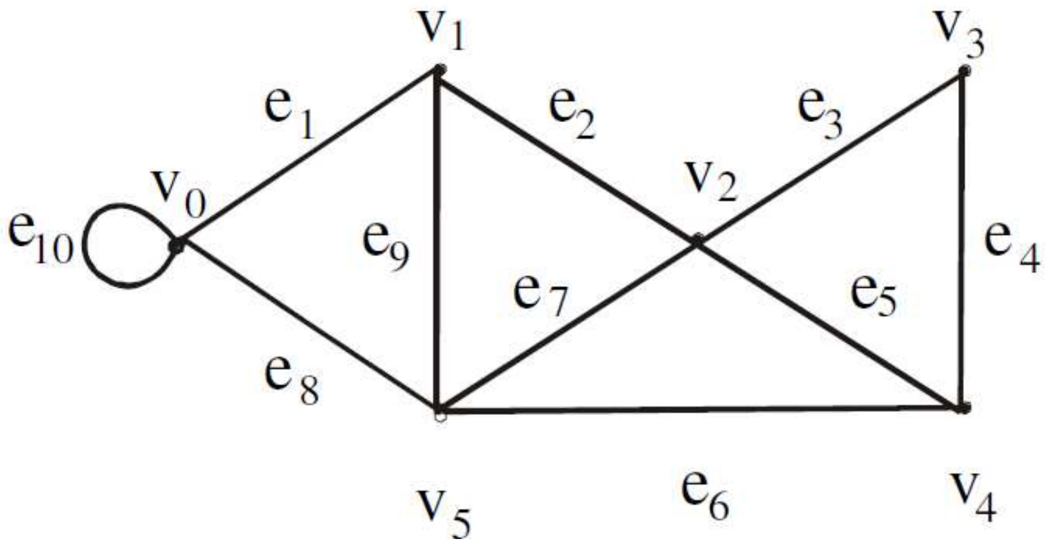
GRAPH CONNECTIVITY

	Repeated Edge	Repeated Vertex	Starts and Ends at Same Point
walk	allowed	Allowed	allowed
closed walk	allowed	Allowed	yes(means, where it starts also ends at that point)
circuit	no	Allowed	yes
simple circuit	no	first and last only	yes
path	no	Allowed	allowed
simple path	no	no	No

GRAPH CONNECTIVITY

- **EXERCISE:**
- In the graph below, determine whether the following walks are paths, simple paths, closed walks, circuits, simple circuits, or are just walks.

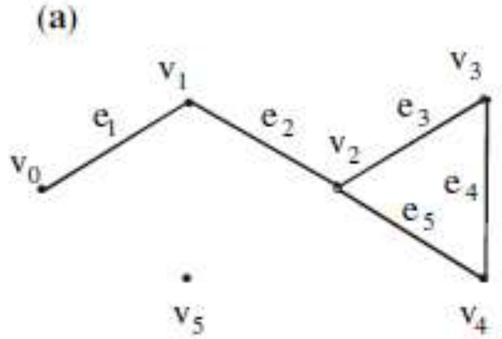
(a)	$v_1e_2v_2e_3v_3e_4v_4e_5v_2e_2v_1e_1v_0$
(b)	$v_1v_2v_3v_4v_5v_2$
(c)	$v_4v_2v_3v_4v_5v_2v_4$
(d)	$v_2v_1v_5v_2v_3v_4v_2$
(e)	$v_0v_5v_2v_3v_4v_2v_1$
(f)	$v_5v_4v_2v_1$



WALK

SOLUTION:

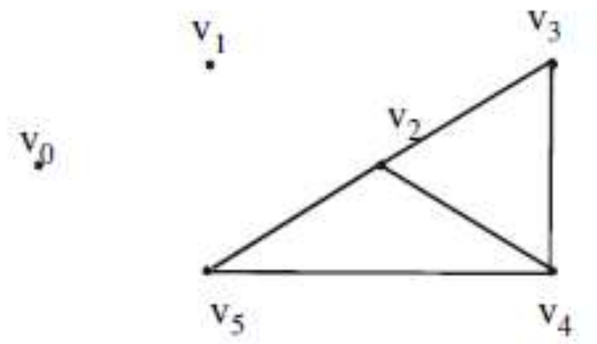
(a) $v_1 e_2 v_2 e_3 v_3 e_4 v_4 e_5 v_2 e_2 v_1 e_1 v_0$



This graph starts at vertex v_1,then goes to v_2 along edge e_2,and moves continuously, at the end it goes from v_1 to v_0 along e_1.Note it that the vertex v_2 and the edge e_2 is repeated twice, and starting and ending, not at the same points.Hence The graph is just a walk.

CLOSED WALK

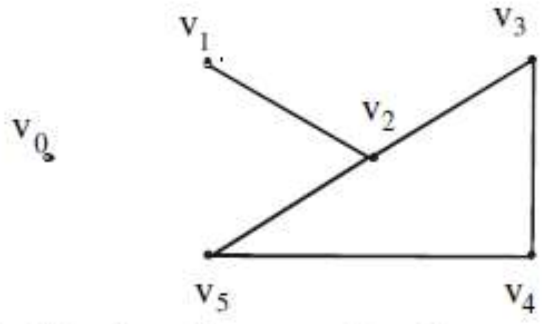
(c) $v_4v_2v_3v_4v_5v_2v_4$



As vertices v_2 & v_4 are repeated and graph starts and ends at the same point v_4,also the edge(i.e. e_5)connecting v_2 & v_4 is repeated, so the graph is a closed walk.

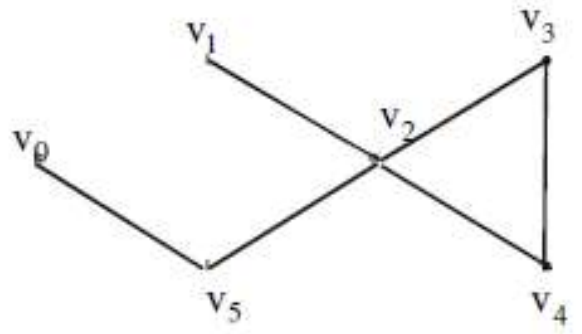
PATH

(b) $v_1v_2v_3v_4v_5v_2$



In this graph vertex v_2 is repeated twice.As no edge is repeated so the graph is a path.

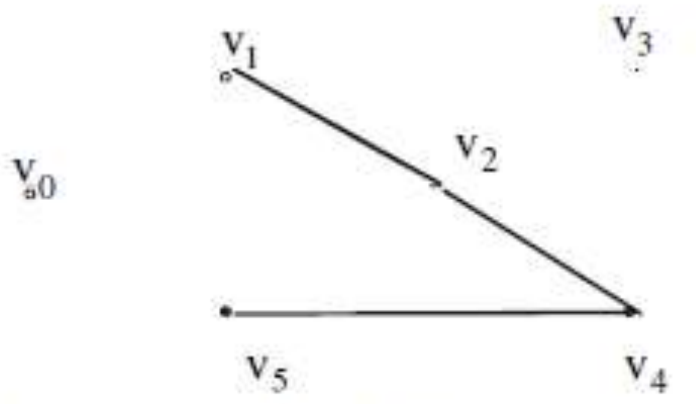
(e) $v_0v_5v_2v_3v_4v_2v_1$



Here vertex v_2 is repeated and no edge is repeated so the graph is a path.

SIMPLE PATH

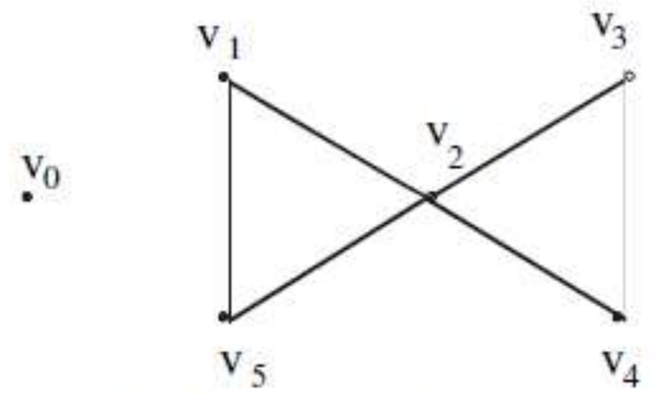
(f) $v_5v_4v_2v_1$



Neither any vertex nor any edge is repeated so the graph is a simple path.

CIRCUIT

(d) $v_2v_1v_5v_2v_3v_4v_2$



In this graph, vertex v_2 is repeated and the graph starts a nd end at the same vertex (i.e. at v_2) and no edge is repeated, hence the above graph is a circuit.